**memory map**

Gameduino uses 4K memory, organised into different functions. The background section controls background colors and sprites. The sprite section controls the foreground sprite graphics.

**screen**

Sprite RAM is a grid of 64x64 bytes, each byte is a single character index. This byte controls the character image and palette used for that 8x8-pixel cell. The total size of the background screen is 512x512 pixels, but only 256x256 pixels are visible. The SPR_X and SCROLL_Y registers control the position of this 400x400 pixel window within the larger screen area.

**characters**

Characters are 8x8 grids of pixels, defined by the values in the character data and palette RAMs. The character data RAM holds the 54 pixels of the character image, encoded using two bits per pixel. The hardware uses these two bits to look up the final color in the character’s 4-entry palette.

For example, a character with a palette of blue, yellow, red and white might appear as shown below. In the left-hand square, the pixel values are shown. In the middle squares, these pixel values are binary listed. In the right-hand column are the hex values, as they appear in memory for this character.

```
0 1 1 1 1 1 1 1
0 0 1 1 0 1 0 1
0 0 0 1 1 1 1 1
0 0 0 0 1 1 1 1
```

**sprite control**

Sprites: these 16x16 pixel sprites can cover any area on the screen. Sprites are drawn from back-to-front, so higher-numbered sprites cover up lower-numbered ones. Each sprite’s appearance is controlled by a 16-bit word.

**sprite palette select**

Each pixel of the sprite image is indexed and looked up in a sprite palette. This palette is a list of colors. Gameduino gives you several palette options: a 256-color palette, a 16-color palette, or four images in 2-bit mode (4 color palette).

**sprite rotate**

Each sprite has a 6-bit ROT field that applies a simple rotation and flip to the sprite image.

**sprite collision detection**

The game is designed to detect when sprites overlap. As it draws the image, Gameduino keeps track of which pixels cover others, and writes the results to the collision RAM.

If each byte in the collision RAM corresponds with the same character index in the character data RAM, the byte value is 0xFF. If the sprite covers any part of another sprite, the value is the index of the other sprite. For example, if sprites 00–03 are arranged like this:

```
0 1 2 3
2 3 1 0
```

The collision RAM looks like this:

```
0x0000 0x0002 0x0002 0x0002
0x0002 0x0002 0x0002 0x0002
0x0002 0x0002 0x0002 0x0002
0x0002 0x0002 0x0002 0x0002
```

**characters**

Characters are 8x8 grids of pixels, defined by the values in the character data and palette RAMs. The character data RAM holds the 54 pixels of the character image, encoded using two bits per pixel. The hardware uses these two bits to look up the final color in the character’s 4-entry palette.

For example, a character with a palette of blue, yellow, red and white might appear as shown below. In the left-hand square, the pixel values are shown. In the middle squares, these pixel values are binary listed. In the right-hand column are the hex values, as they appear in memory for this character.

```
0 1 1 1 1 1 1 1
0 0 1 1 0 1 0 1
0 0 0 1 1 1 1 1
0 0 0 0 1 1 1 1
```

**quick reference**

For a quick reference, see the quick reference section at the end of this document.

**copyright**

©2011 gameduino.com